

Resolución segundo parcial 1c2023

Santiago Cifuentes

1. Enunciado

- 1) Sea $G = (V, E)$ un grafo pesado con pesos positivos y $D \subseteq V$ un subconjunto de sus nodos, a los cuales llamaremos *depósitos*. Dado un nodo cualquiera v , decimos que su *distancia a los depósitos* es el mínimo entre las distancias de v a cada depósito $d \in D$, y lo notamos como $d_G(D, v)$. Es decir, $d_G(D, v) = \min\{d_G(w, v) : w \in D\}$.

- a) Proponer un algoritmo que calcule la distancia a los depósitos de todos los nodos, justificando su diseño. Complejidad esperada: $O(n^2)$.

Decimos que una arista es *inútil* si **ningún camino mínimo** de los depósitos a los vértices usa esa arista.

- b) Proponer un algoritmo que detecte las aristas inútiles, justificando su diseño. Complejidad esperada: $O(n^2)$.

Denotamos como $\maxDist(G, D)$ a la máxima distancia de un nodo a los depósitos. Dicho formalmente, $\maxDist(G, D) = \max\{d_G(D, v) : v \in V \setminus D\}$. Queremos agregar al grafo una arista e de un cierto costo positivo q de manera tal de minimizar $\maxDist(G \cup \{e\})$ ¹. Esta arista no puede estar conectada a ningún depósito.

- c) Proponer un algoritmo que decida, dado el costo positivo q , qué arista se debe agregar al grafo para minimizar la máxima distancia a los depósitos. Aparte, indicar cuánto mejora esta máxima distancia. Complejidad esperada: $O(n^3)$.

Criterio de aprobación: Dos de los tres incisos deben estar bien resueltos.

- 2) En la Facultad de Ciencias Naturales y Exactas (FCNyE) se dictan M materias. Algunas deben cursarse en simultáneo, y por lo tanto al conjunto de todas las materias se lo particiona en k conjuntos disjuntos G_1, \dots, G_k con la intención de que les estudiantes cursen un grupo G_i por cuatrimestre.

Todas las materias deben tomar una evaluación en la última semana del cuatrimestre considerando las siguientes restricciones:

- Las evaluaciones de cada materia deben hacerse en los días en que se dicta la materia.
- No puede haber dos evaluaciones de un grupo G_i en un mismo día.
- Solo hay A aulas, por lo que cada día se pueden hacer a lo sumo A evaluaciones.

El Secretario Académico quiere saber si es factible armar una asignación de instancias de evaluación para que todas las materias puedan tomar sus evaluaciones en la última semana del cuatrimestre.

Nota: Suponer que la semana tienen 6 días hábiles, y que se cuenta con una función $DiasCursada(m)$ que devuelve, dada una materia m , una lista con los días de la semana en que se cursa la materia m ².

Se pide:

- a) Modelar el problema de asignación como un problema de flujo.
- b) Dar una interpretación a cada unidad de flujo y cada restricción de capacidad.
- c) Justificar que el modelo es correcto. En particular, mostrar cómo se reconstruye una asignación válida en caso de que exista.
- d) Determinar la complejidad temporal del algoritmo en función del tamaño de la entrada. Asumir que para resolver el modelo de flujo se emplea el Algoritmo de Edmonds y Karp, y dar una cota ajustada. La entrada del algoritmo consiste en la partición G_1, \dots, G_k , el número A y la estructura de datos que responde las consultas de $DiasCursada(\cdot)$ de tamaño $O(M)$.

Criterio de aprobación: El modelado tiene que ser correcto, y se debe explicar claramente su diseño y su tamaño (en función de los parámetros de entrada).

Ayuda: Representar con un nodo a cada par (G_i, d) con G_i el grupo i y $1 \leq d \leq 6$ el día de la semana.

¹En un abuso de notación $G \cup \{e\}$ denota el grafo que se obtiene de agregar a G la arista e

²Se puede asumir que esta función devuelve la lista en $O(1)$.

3) Dados los problemas

Π_1 : ÁRBOL GENERADOR DE GRADO ACOTADO

Entrada: grafo H ; $k \in \mathbb{N}$

Pregunta: ¿tiene H un árbol generador T tal que $d_T(v) \leq k$ para todo nodo v de T ?

Π_2 : CAMINO HAMILTONIANO

Entrada: grafo G .

Pregunta: ¿existe un camino que pasa exactamente una vez por cada vértice de G ?

queremos probar que Π_1 es NP-COMPLETO asumiendo que Π_2 es NP-HARD.

- a) Demostrar que Π_1 pertenece a NP.
- b) Demostrar que Π_1 es NP-HARD.
- c) Demostrar que Π_1 es NP-COMPLETO.
- d) ¿Es cierto que Π_1 es al menos tan difícil como **cualquier** problema que pertenece a la clase NP? ¿El problema se vuelve más sencillo si el valor k está fijo? Justificar.

Criterio de aprobación: La reducción del inciso b) tiene que ser correcta, y le estudiante debe mostrar un manejo claro de los conceptos involucrados.

2. Resolución

1. Ejercicio 1:

- a) Nos piden calcular, para cada v , $d_G(D, v) = \min\{d_G(w, v) : w \in D\}$. Notemos que por definición dado un v cualquiera podemos obtener este valor en $O(\min\{m \log n, n^2\}) = O(n^2)$ usando Dijkstra³: calculamos todas las distancias $d(v, z)$ calculando el camino mínimo de 1 a todos desde v y luego tomamos el mínimo. Si hacemos esto para cada nodo obtenemos un algoritmo $O(n^3)$ que resuelve el problema, pero esta complejidad es mayor que la pedida. Podemos mejorar un poco si observamos que alcanza con calcular las distancias desde los depósitos, y no desde cualquier nodo, por lo que usamos Dijkstra solamente desde los nodos de D . Esto induce un algoritmo $O(|D| \times n^2)$, que también supera la complejidad pedida.

La observación clave es que no necesitamos conocer todas las distancias desde los nodos de D , sino que únicamente estamos interesados en la menor para cada nodo $v \in V \setminus D$. Intuitivamente queremos calcular el camino mínimo desde un conjunto de nodos a todos, y este es un problema que ya vimos en el contexto de BFS.

Nos gustaría que la búsqueda de Dijkstra comience “desde todos los depósitos a la vez”. Para modelar este comportamiento vamos a agregar un nodo v_0 al grafo, y conectaremos a v_0 con todos los nodos de D con aristas de costo cero. Luego, si ejecutamos Dijkstra desde v_0 estaremos simulando esta idea de ejecutar Dijkstra desde varios nodos a la vez.

Formalicemos esto último: siendo G' el grafo obtenido al agregar v_0 y los ejes de v_0 a los depósitos con costo 0, probemos que $d_G(D, v) = d_{G'}(v_0, v)$. Probemos las dos desigualdades:

\leq) Si $d_{G'}(v_0, v) = k$ entonces hay un camino de costo k de v_0 a v en G' . Como v_0 solo está conectado a los depósitos, este camino tiene que ser de la forma $P = v_0 d_v \dots v$ con $d_v \in D$. Notemos que⁴ $c(P) = c(v_0 d_v) + c(d_v \dots v) = c(d_v \dots v)$ porque $c(v_0, d_v) = 0$ por construcción, y el camino $d_v \dots v$ está contenido en G ya que de G' tiene los mismos nodos y ejes que G salvo por v_0 , pero este nodo no es parte de esta porción del camino. Luego, hay un camino de un depósito a v de costo k , y por lo tanto $d_G(D, v) \leq k$.

\geq) Si $d_G(D, v) = k$ entonces hay un depósito d_v y un camino $d_v \dots v$ en G con costo k . Notemos entonces que $v_0 d_v \dots v$ es un camino de costo k de v_0 a v en G' . Por lo tanto, $k \geq d_{G'}(v_0, v)$.

Con esta prueba reducimos el problema de calcular $d_G(D, v)$ al de calcular $d_{G'}(v_0, v)$. Podemos computar esto para cada v haciendo Dijkstra desde v_0 .

Suponiendo que el grafo viene representado como una lista de adyacencias, el algoritmo final entonces es:

- 1) Construimos en $O(n)$ a G' agregando a G un nodo v_0 y aristas $v_0 d$ hacia cada depósito d con costo 0.

³El cual podemos usar porque los costos son todos positivos.

⁴Voy a denotar el costo de la arista vw como $c(vw)$, y con un abuso de notación voy a extender esto a caminos.

- 2) Ejecutamos Dijkstra desde v_0 usando la implementación para grafos densos, lo que tomará $O((n+1)^2) = O(n^2)$ operaciones.
- 3) Devolvemos en $O(n)$ las distancias calculadas por Dijkstra.

Este algoritmo es correcto por lo que probamos, y tiene una complejidad final de $O(n^2)$.

- b) Para el inciso b) podemos diseñar un criterio similar a los de la guía para identificar aristas **útiles** (es decir, que se usan en algún camino mínimo) en base a las distancias de los nodos a los depósitos (que ya sabemos obtener en $O(n^2)$ gracias al inciso a)). Notemos que si una arista vw es útil entonces hay un camino mínimo de los depósitos a algún nodo z que la usa. Pero, en particular, ese camino mínimo debe contener un camino mínimo hacia w y hacia v . Por lo tanto, suponiendo que la arista se usa con la orientación vw , la conclusión sería que $d(D, w) = d(D, v) + c(vw)$.

Probemos este lema: una arista vw es **útil** si y solamente si $d(D, w) = d(D, v) + c(vw)$ o $d(D, v) = d(D, w) + c(vw)$.

\Rightarrow) Si una arista vw es útil entonces es parte de un camino mínimo P de los depósitos a un nodo z . Si suponemos sin pérdida de generalidad que vw se usa con esa orientación, entonces $P = d_z \dots vw \dots z$ donde d_z es un depósito. Como P es un camino mínimo de d_z a z la porción del camino que va de d_z a v y la que va de d_z a w también tienen que ser un camino mínimo de los depósitos a v y w , respectivamente. Luego, concluimos que $d(D, v) + c(vw) = d(D, w)$.

\Leftarrow) Por definición, si $d(D, v) + c(vw) = d(D, w)$ entonces vw pertenece a un camino mínimo de los depósitos a w (el camino es exactamente el camino mínimo hasta v más la arista vw).

Con este lema construimos el siguiente algoritmo:

- 1) Calculamos en $O(n^2)$ todas las distancias $d(D, v)$.
- 2) Por cada arista vw verificamos las igualdades de nuestro lema. Si no cumple ninguna de las dos entonces la marcamos como inútil. Hay que procesar m aristas, y como tenemos las distancias precalculadas cada verificación nos toma $O(1)$.

Es decir, este algoritmo tiene complejidad $O(n^2 + m) = O(n^2)$ ya que $m \leq n^2$ en cualquier grafo simple.

- c) Si C es el conjunto de aristas que podemos usar, entonces nos piden encontrar $\min_{e \in C} \{ \maxDist(G \cup \{e\}) \}$, y aparte identificar qué arista se usa para encontrar el mínimo. Notemos que dada e es posible calcular $\maxDist(G \cup \{e\})$ en $O(n^2)$ usando el inciso a): simplemente se calculan todas las distancias de los nodos a los depósitos y luego nos quedamos con el máximo. Esto induce un algoritmo $O(|C| \times n^2)$ (por cada arista calculamos el valor) que es $O(n^4)$ si $C = \Theta(n^2)$, por lo que no alcanza la complejidad pedida.

Vamos a mantener esta idea de recorrer todas las aristas, pero intentemos calcular más rápido el valor $\maxDist()$. Notemos que si logramos calcularlo en $O(n)$ entonces la complejidad final del algoritmo será $O(|C| \times n) = O(n^3)$.

Sea $G' = G \cup \{xy\}$. Para calcular $\maxDist(G')$ alcanza con encontrar todas las distancias de los nodos a los depósitos. Dado un nodo v cualquiera hay dos opciones para su camino mínimo a D : este usa la arista nueva e , o bien no la usa. Si no la usa, entonces $d_{G'}(D, v) = d_G(D, v)$, mientras que si la usa entonces el camino mínimo a los depósitos es de la forma $d_v \dots xy \dots v$ o bien $d_v \dots yx \dots v$, donde d_v es un depósito. En ambos casos los subcaminos del camino tienen que ser mínimos, y por lo tanto podemos pensar que el camino $P = d_v \dots xy \dots v$ se descompone en $P = (d_v \dots x) + xy + (y \dots v) = P_1 + xy + P_2$ donde P_1 es un camino mínimo de los depósitos a x y P_2 es un camino mínimo de y a v . Aparte, estos caminos P_1 y P_2 no usan la arista nueva, por lo que son caminos mínimos del grafo original. Luego, si el camino mínimo de v a los depósitos usa la arista xy entonces este camino debe tener costo $\min(d_G(D, x) + c(xy) + d_G(y, v), d_G(D, y) + c(yx) + d_G(x, v))$.

Con estas observaciones probamos lo siguiente: $d_{G'}(D, v) = \min\{d_G(D, v), d_G(D, x) + c(xy) + d_G(y, v), d_G(D, y) + c(yx) + d_G(x, v)\}$. Es decir, para calcular la distancia de un nodo a los depósitos en G' alcanza con usar las distancias del grafo original y hacer algunas pocas cuentas. Si tenemos precalculadas estas distancias, entonces podemos encontrar la distancia de un nodo a los depósitos en $O(1)$. Observemos que hacen falta las distancias de todos los nodos a todos los nodos.

Esto nos permite mejorar el algoritmo de búsqueda exhaustiva que propusimos al principio de la resolución:

- 1) Primero calculamos todas las distancias a los depósitos $d(D, v)$ y las distancias entre los nodos $d(v, w)$. Esto toma $O(n^2)$ usando el inciso a) y $O(n^3)$ usando el algoritmo de Floyd, respectivamente.
- 2) Por cada arista $xy \in C$ calculamos $\maxDist(G \cup \{xy\})$, y nos quedamos con la arista que minimice este valor. Para calcular $\maxDist(G \cup \{xy\})$ usamos que

$$\maxDist(G \cup \{xy\}) = \max_{v \in V} \{d_{G \cup \{xy\}}(D, v)\} =$$

$$= \max_{v \in V} \{\min\{d_G(D, v), d_G(D, x) + c(xy) + d_G(y, v), d_G(D, y) + c(yx) + d_G(x, v)\}\}$$

Es decir, por cada nodo solo tenemos que hacer algunas comparaciones basadas en las distancias que precalculamos antes, por lo que el cálculo de cada $\maxDist(G \cup \{xy\})$ requiere $O(n)$ operaciones.

En total, este paso toma $O(|C| \times n) = O(mn)$.

El algoritmo completo es correcto por las observaciones que hicimos antes, y tiene complejidad $O(n^3 + mn) = O(n^3)$, que cumple con la consigna.

2. Ejercicio 2:

- a) Este es un problema de asignación de materias a días, donde hay algunas condiciones adicionales sobre cómo debe ser esta asignación. En principio, seguro que nuestra red va a tener que representar a los días por un lado y a las materias por otro.

Notemos primero que no podemos conectar una materia m a cualquier día, sino que debemos respetar sus días de cursada. A su vez, si conectamos cada materia a sus días válidos no estaríamos codificando la condición de los **grupos**: por cada grupo G_i puede haber una única materia tomando evaluación en el día d .

Para agregar esta condición vamos a “filtrar” todas las materias de un mismo grupo que se cursen en un mismo día, de tal forma de permitir que solo una se curse por cada día. El modelo se puede ver en la siguiente figura:

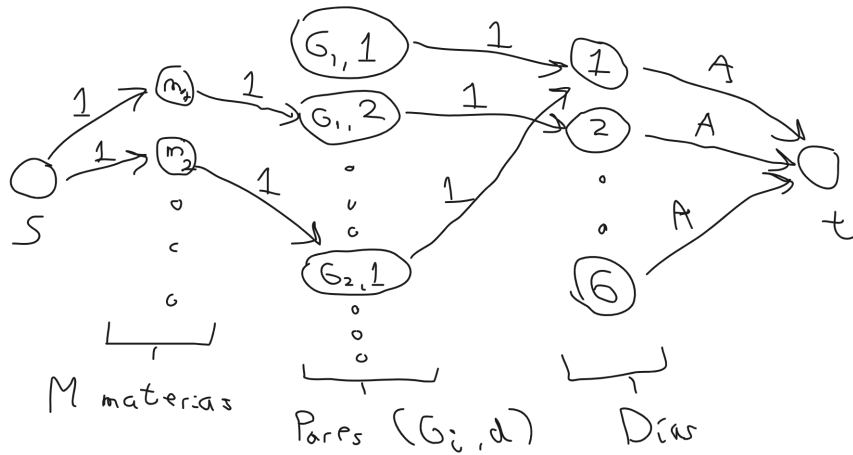


Figura 1: Red del ejercicio 2). En este caso se ve que $m_1 \in G_1$, $m_2 \in G_2$, $2 \in DiasCursada(m_1)$ y $1 \in DiasCursada(m_2)$

En nodo s de la red se conecta con M nodos materia, teniendo cada eje capacidad 1, ya que cada materia debe tomar una única evaluación. Luego, cada nodo materia m se conecta con un eje de capacidad 1 a todos los nodos de la forma (G_i, d) con $m \in G_i$ y $d \in DiasCursada(m)$. Es decir, el nodo (G_i, d) va a recibir flujo de aquellas materias del grupo G_i que se cursan el día d , y la idea es que solo puede permitir que una sola de esas unidades de flujo pase. Finalmente, unificamos las materias que se cursan en el día d en el nodo que representa el día d : este tiene un eje entrante de capacidad 1 desde cada nodo de la forma (G_i, d) . Como puede haber a lo sumo A evaluaciones por día se agrega el eje de cada día d hacia t con capacidad A .

Proponemos que será posible que todas las materias tomen su evaluación siguiendo las restricciones si y solamente si hay un flujo de valor M en la red.

- b) Una unidad de flujo que va de s a t hace un camino de la forma $s m (G_i, d) d t$. Este camino representa que la materia m que pertenece al grupo G_i toma evaluación en el día d . Por como está construido el grafo la materia m pertenece efectivamente al grupo G_i y puede tomar evaluación en el día i .

- c) Para probar la correctitud, veamos que existe una asignación de materias a días que permite tomar U evaluaciones si y solamente si hay un flujo de valor U en la red R que definimos antes.
- \Rightarrow) Sea $S = \{(m_i, d_i)\}$ una asignación de materias válida con $|S| = U$. Vamos a definir un flujo válido en la red R :

- ★ Por cada $(m_i, d_i) \in S$ ponemos una unidad de flujo en el camino $s m_i (G_j, d_i) d_i t$ con G_j el grupo de m_i .

Notemos que por construcción y porque S es una asignación válida los ejes que nombramos en ese camino existen.

Veamos primero que se respetan todas las capacidades:

- Ejes $s m_i$: como S es una asignación válida ninguna materia está asignada más de un día. Luego, por cada uno de estos ejes pasa a lo sumo una unidad de flujo.
- Ejes $m_i (G_j, d)$: de nuevo, cada materia está asignada a lo sumo a un día, y por lo tanto por estos ejes pasa a lo sumo una unidad de flujo.
- Ejes $(G_j, d) d$: si pasara más de una unidad de flujo por uno de estos ejes entonces la asignación S estaría asignado a dos materias de un mismo grupo el mismo día de evaluación, lo cual es absurdo.
- Ejes $d t$: como S es una asignación válida en cada día se toman a lo sumo A parciales. Por como armamos el flujo, las unidades que salen de d son las mismas que la cantidad de parciales que se toman en el día d , y este valor es menor o igual a A .

Por otro lado, es trivial ver que el flujo se conserva porque lo definimos en base a caminos de s a t .

Concluimos aparte que el valor del flujo es U , ya que salen U unidades de flujo de s (uno por cada asignación de S).

\Leftarrow) Sea f un flujo válido en R de valor U . Vamos a armar una asignación válida de materias a días que permita tomar U evaluaciones. Definimos la asignación de la siguiente forma:

- ★ Si por la arista $m_i (G_j, d)$ pasa una unidad de flujo, entonces m_i toma su evaluación el día d .

Veamos que esta asignación es válida:

- *Cada parcial toma a lo sumo una evaluación*: a m_i puede entrar a lo sumo una unidad de flujo, y, por conservación, solo puede salir una. Por lo tanto, a cada materia le asignamos a lo sumo un día.
- *Las materias toman su evaluación en sus días de cursada*: por construcción, le asignamos el día d a la materia m_i si hay un eje $m_i (G_j, d)$ en la red por el que pasa una unidad de flujo. Esto solo puede ocurrir, por construcción, si m_i se cursa en el día d .
- *Cada día se toman a lo sumo A evaluaciones*: la cantidad de evaluaciones que se toman en el día d es igual a la cantidad de flujo que llega a d , dado que si hay una unidad de flujo en la arista $m_i (G_j, d)$ entonces esa unidad llega luego a d . El flujo que sale de d está acotado por A , y entonces por conservación también el que entra. Luego, asignamos a lo sumo A evaluaciones al día d .
- *Cada grupo de materias toma a lo sumo una evaluación por día*: dos materias toman evaluación el mismo día si hay dos aristas $m_1 (G_j, d)$ y $m_2 (G_j, d)$ ambas con una unidad de flujo. Pero desde (G_j, d) solo puede salir una unidad de flujo, y entonces por conservación solo puede entrar una. Concluimos que no puede haber dos materias de un mismo grupo tomando evaluación un mismo día.

Como el valor del flujo era U , la asignación de materias permite tomar U evaluaciones.

Gracias a esta prueba podemos decir lo siguiente: hay una asignación válida que permite que todas las materias tomen su evaluación si y solamente si hay un flujo de valor M en la red.

- d) Para estudiar la complejidad de resolver el modelo tenemos que calcular su tamaño y estimar una cota para el flujo:
- Nodos: son $1 + M + K \times 6 + d + 1$ (s , materias, pares grupo-día, días y t). Esto es $O(M)$.
 - Ejes: son a lo sumo $M + M \times 6 + K \times 6 + d$ (de s a las materias, de las materias a sus pares grupo-día, de cada par grupo-día a un único día y de los días a t). Esto es $O(M)$.
 - Flujo: es a lo sumo M (se puede ver tomando el corte que se queda con s por un lado y el resto de la red por otro).

Con estos parámetros podemos escribir la complejidad final del algoritmo de Edmonds y Karp aplicado a esta red:

$$O(\min\{mF, nm^2\}) = O(\min\{M^2, M^3\}) = O(M^2)$$

3. Ejercicio 3:

- a) Para probar que $\Pi_1 \in \text{NP}$ hay que mostrar que las instancias positivas admiten un certificado de tamaño polinomial capaz de ser verificado por un certificador polinomial.

Por definición, $\langle H, k \rangle \in \Pi_1$ si y solamente si existe un árbol generador T de H de grado acotado por k . Luego, este podría ser nuestro certificado: notemos que T tiene tamaño polinomial en función de H , y que este certificado existe únicamente si la instancia es positiva. Definamos al certificador M .

M recibe al grafo H y al certificado T . Suponiendo que T representa un grafo válido⁵, M tiene que:

- Verificar que $T \subseteq H$, que se hace en $O(n^2)$ suponiendo que ambos vienen representados como matriz de adyacencias.
- Verificar que T es un árbol (i.e. que no tiene ciclos y es conexo), lo cual requiere $O(n^2)$ operaciones sobre la matriz empleando DFS.
- Verificar que todos los nodos tienen grado acotado por k , lo cual también requiere un recorrido lineal sobre la matriz en $O(n^2)$.

Por lo tanto el verificador realiza una cantidad de operaciones polinomial en función del tamaño de entrada, y concluimos que $\Pi_1 \in \text{NP}$.

- b) Para ver que Π_1 es NP-HARD alcanza con probar que otro problema NP-HARD se reduce a él. Luego, por transitividad, se deduce que todos los problemas $\Pi \in \text{NP}$ se reducen a Π_1 .

Como podemos asumir que Π_2 es NP-HARD alcanza con probar que $\Pi_2 \leq_p \Pi_1$. Es decir: tenemos que encontrar una función polinomial f que tome entradas válidas de Π_2 (i.e. grafos) y devuelva entradas válidas de Π_1 (i.e. grafos junto a un número natural k) y que cumpla

$$G \in \text{HAM} \iff f(G) = \langle H, k \rangle \in \text{AGGA}$$

Probemos que $f(G) = \langle G, 2 \rangle$ funciona. Es decir, tenemos que ver que:

- Es polinomial.
- Lleva instancias positivas a positivas y negativas a negativas.

Lo primero es trivialmente cierto (la función solo arma el par con el 2 en la segunda componente). Ahora nos queda probar la “correctitud”.

La idea de la función es la siguiente: si $G \in \text{HAM}$ entonces G contiene un camino hamiltoniano, y este es exactamente un árbol generador de grado acotado 2. Para la vuelta deberíamos mostrar que todo árbol generador acotado de grado 2 es a su vez un camino hamiltoniano.

Probemos la doble implicación $G \in \text{HAM} \iff f(G) = \langle G, 2 \rangle \in \text{AGGA}$:

\implies) Si $G \in \text{HAM}$ entonces G contiene un camino hamiltoniano $P = v_1 \dots v_n$. Notemos que P es literalmente un árbol generador de grado acotado 2: P es un subgrafo conexo que no contiene ciclos (por ser un **camino simple**⁶) y aparte contiene a todos los nodos (por ser **hamiltoniano**). Aparte, cada nodo tiene grado a lo sumo 2. Concluimos entonces que $\langle G, 2 \rangle \in \text{AGGA}$.

\impliedby) Si $f(G) = \langle G, 2 \rangle \in \text{AGGA}$ entonces G tiene un árbol generador T cuyos nodos tienen todos grado acotado por 2. Mostremos que T es un camino hamiltoniano:

Sean v, w dos hojas distintas de T ⁷. Como T es un árbol hay un único camino P entre v y w . Notemos que todos los nodos del árbol tienen que estar en P : si hay un nodo $z \notin P$ entonces puedo considerar a w , el nodo de P más cercano a z . Este nodo w es parte del camino de P , por lo que tiene dos vecinos w_1 y w_2 que son parte del camino. Pero aparte resulta que tiene un tercer vecino w_3 que es parte del camino hacia z (notemos que $w_3 \neq w_1, w_2$ porque elegí a w como el nodo más cercano a z). Esto implica que el grado de w en T es al menos 3, lo cual es absurdo.

Por lo tanto, P es un camino que recorre todos los nodos de G , y entonces G tiene un camino hamiltoniano y $G \in \text{HAM}$.

- c) Un problema es NP-COMPLETO si es NP y NP-HARD. En a) probamos que Π_1 es NP, y en b) que Π_1 es NP-HARD. Luego, Π_1 es NP-COMPLETO.
- d) Si entendemos “fácil” como “admite un algoritmo polinomial” entonces Π_1 es tan difícil (entendiendo a difícil como al complemento de ser fácil) como cualquier otro problema NP, dado que al ser NP-COMPLETO un algoritmo polinomial para resolver Π_1 podría adaptarse a un algoritmo polinomial

⁵En principio el certificado T es solo una secuencia de bits, por lo que M debería corroborar que T represente realmente un grafo. Esto lo puede hacer comprobando que T sea una matriz de adyacencias, por ejemplo.

⁶Por las dudas, todos los caminos hamiltonianos son simple por definición.

⁷Acá estoy usando el lema de que todo árbol con más de un nodo tiene dos hojas, que se puede probar fácilmente por inducción y es un ejercicio de las primeras guías.

para resolver cualquier otro problema de la clase NP . En particular, también vale la vuelta: si existiera un algoritmo polinomial para resolver algún problema NP-COMPLETO (como por ejemplo 3-COLOREO) entonces se podría adaptar ese algoritmo para resolver Π_1 .

Notemos que el problema de AGGA no se vuelve más fácil si k está fijo. En particular, la demostración del inciso c) llevaba instancias de hamiltoniano a instancias de AGGA donde k siempre valía 2, y por lo tanto esa reducción demuestra que el problema es NP-HARD incluso si k está fijo y vale 2. Por otro lado, si $k = 1$ el problema es trivial.